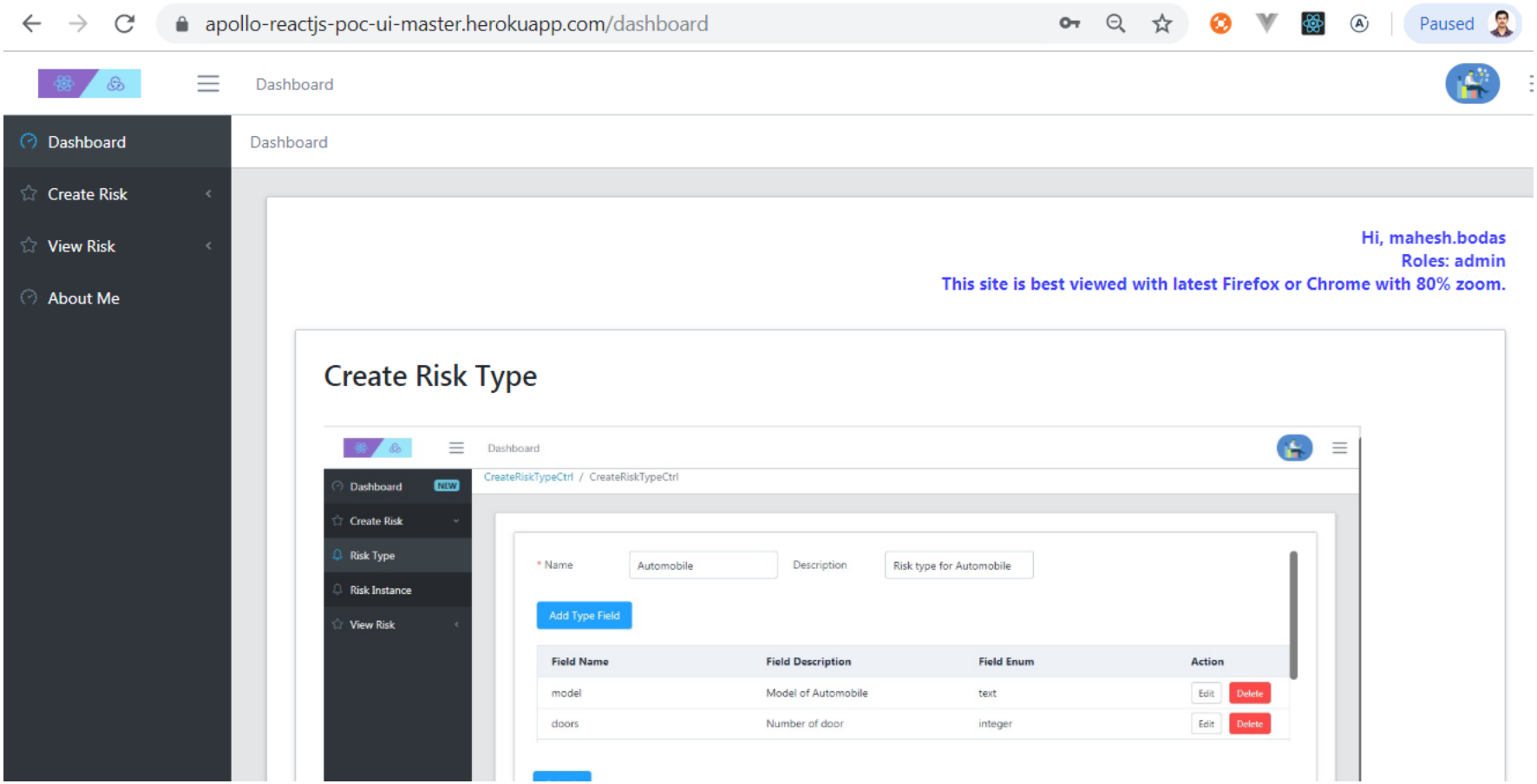
Home Screen


apollo-reactjs-poc-ui-master.herokuapp.com/dashboard
Paused
Dashboard
Dashboard
Create Risk
View Risk
About Me
Dashboard
Hi, mahesh.bodas
Roles: admin
This site is best viewed with latest Firefox or Chrome with 80% zoom.
Create Risk Type
Dashboard
Dashboard
NEW
Create Risk
Risk Type
Risk Instance
View Risk
CreateRiskTypeCtrl / CreateRiskTypeCtrl
* Name
Automobile
Description
Risk type for Automobile
Add Type Field
Field Name
Field Description
Field Enum
Action
model
Model of Automobile
text
Edit
Delete
doors
Number of door
integer
Edit
Delete

Apollo ReactJS PoC UI

- ApolloClient and ReactJS PoC application.
- Use above link to access UI.
- Username :- admin (lower case)
- Password :- poctest#1 (lower case)
- Best viewed with Firefox with 80 % zoom
- Above UI refers Web API at following location credentials are same.
- https://django-poc-server-maheshbodas.herokuapp.com/

Salient features

- Developed using ReactJS, ApolloClient, Redux, redux-thunk, ES6, Element React library, Node JS, Serve as a Static server.
- ApolloClient is use to send GraphQL queries and Mutation to GraphQL based Django server.
- Used Redux as global data store for this application. Redux actions and reducers make is easy to shape and store data as needed by UI.

Salient features (continued)

- Used CoreUI React template so as to reuse Sidebar navigation, Breadcrumbs and common Layout for all pages.

- CoreUI React template itself build using Create React App. CRA is zero configuration template to build, unit test and deploy application on server.

- CRA makes use of Enzyme and Jest for Unit testing. Does not need separate Jest installation or Web pack tweaking. Everything is taken care by CRA.

Salient features (continued)

- Well thought React components to modularize code.
- Made use of Typescript and React Typescript support to develop generic component that’s used in all screen to automate task like showing Page loading Icon, display UI when data is available and show error if any.
- The above component has been reused in all screens of ReactJS PoC application.

Salient features (continued)

- Role based authorization further limits access to screens and components use for creating Risk Types.
- User can define RiskType (Entity Name) and associated RiskTypeFields (Entity attributes) in one go using header and details table in Create Risk Type screen.
- User can add Risk (Entity Instance Name) and Risk Fields (Entity attribute values) in one go using header and dynamically populated controls in Create Risk screen.

Salient features (continued)

- Create Risk Instance screen dynamically adds required field validation and field type specific validation for controls related to Risk type fields. (Entity attributes) i.e. Date / float / integer.

- Successful model creation messages are displayed using MessageBox.

- Display various model validation errors using MessageBox

Salient features (continued)

- View Single Risk display Auto Complete dropdown for selecting Risk Name. Upon selection user clicks Get Risk Detail button to fetch details for given Risk.

- Single Risk Instance data is fetched from GraphQL API.

- Single Risk Instance aggregates Risk Fields as well Field type information. View Single Risk control then dynamically renders appropriate read-only controls based on field type information found in a retrieved Single Risk instance.

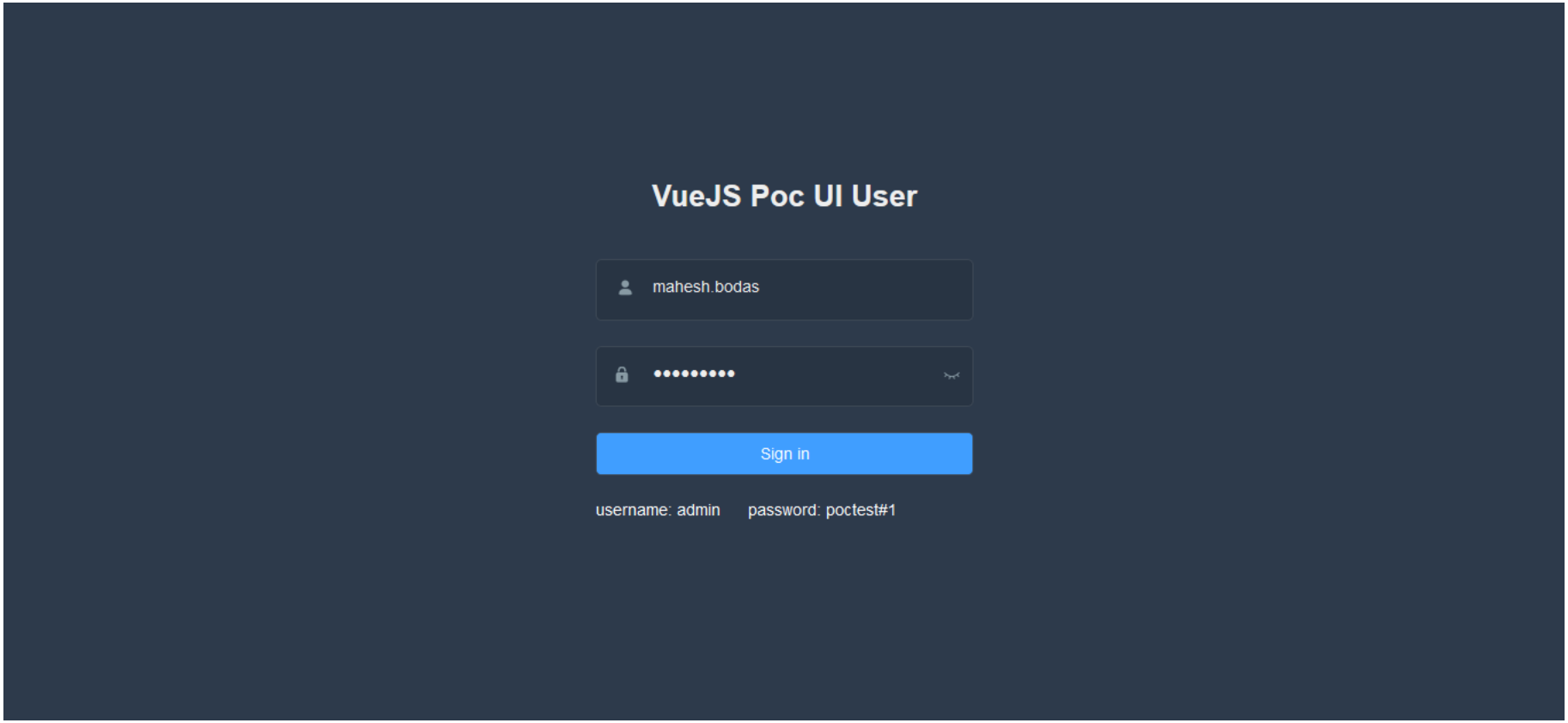
- Each dynamically populated Control’s value is set with corresponding Risk field (Entity attribute) value

Salient features (continued)

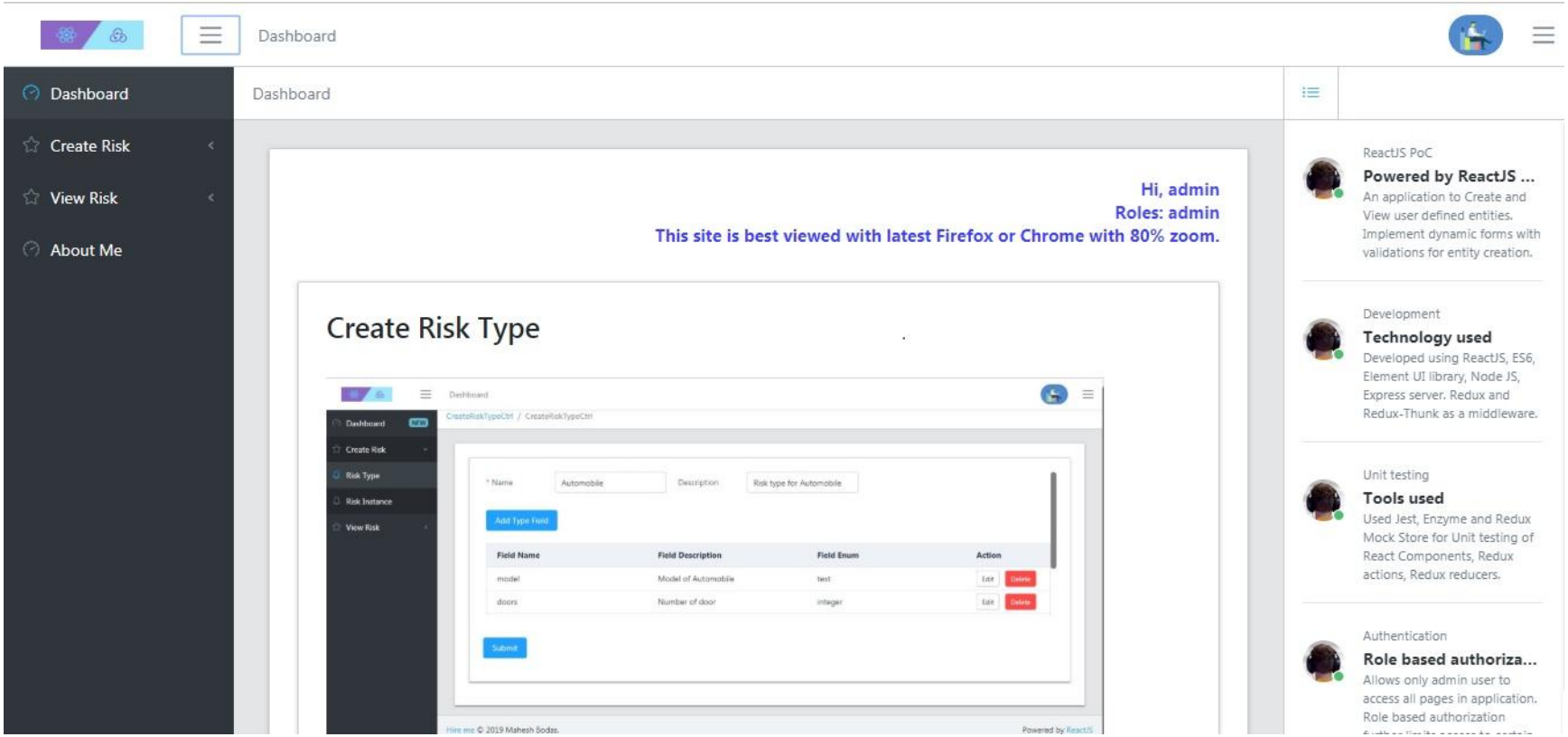
- View All Risk screen allow user to select RiskType from drop down list.
- Upon RiskType selection, screen fetches Risk instances filtered by selected RiskType and allows user to paginate through list of Risk Instances whose foreign key RiskType matches with the Selected RiskType in Dropdown box.

- GraphQL only support forward pagination. Backward pagination is enabled programmatically by maintaining stack in Redux state store.

Login Screen



Dashboard showing app highlights



Create Risk Type screen

- Admin User can define Risk types.
- User Enters Risk type name and description.
- Click Add Risk Type Field button to define Risk type fields (Entity attribute) i.e. Name and Type (String / Date / Integer / Currency).
- Screen allows for editing and deleting Risk Type fields.
- Post Risk type after adding / editing Risk type fields.

Create Risk Type Image

https://reactjs-poc-ui-master.herokuapp.com/create/createrisktypectrl

Paused

Dashboard

- Dashboard
- Create Risk
- Risk Type
- Risk Instance
- View Risk
- About Me

CreateRiskTypeCtrl / CreateRiskTypeCtrl

Field Name	Field Description	Field Enum	Action
housenumber	House number alloted by corporation	text	Edit Delete
floors	Number of floors	integer	Edit Delete
sum	Sum Insurance Amount	currency	Edit Delete
completion	Construction completion date	date	Edit Delete

Submit

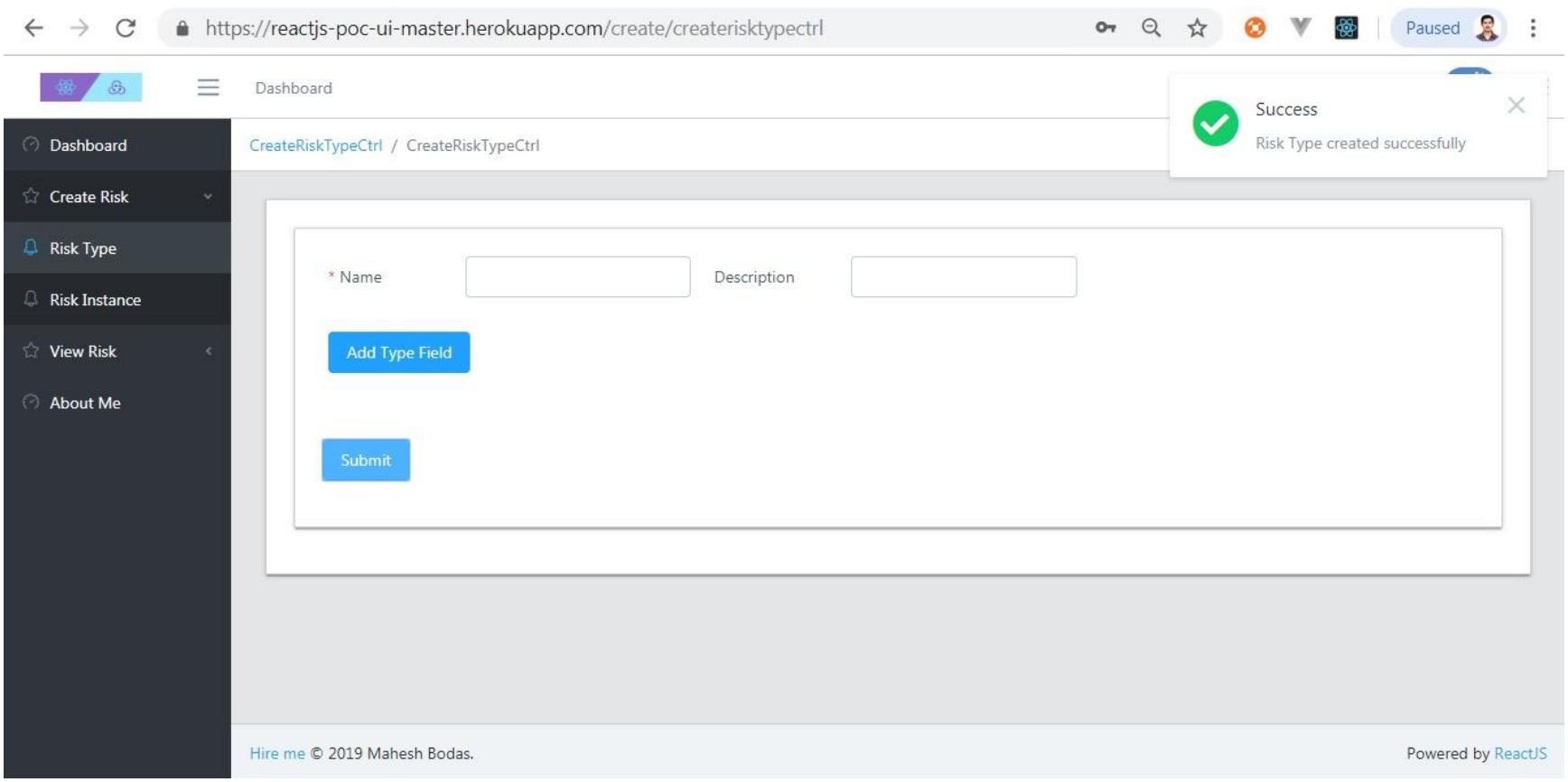
Hire me © 2019 Mahesh Bodas.

Powered by ReactJS

Risk Type Data posted to server


```
{"risk_type_name":"Home","risk_type_description":"Risk type for Home","risktype_risktypefields":[{
"risk_type_field_name":"housenumber","risk_type_field_enum":"text","risk_type_field_description":"House Number"
},{"risk_type_field_name":"floors","risk_type_field_enum":"integer","risk_type_field_description":"Number of
floors"},{"risk_type_field_name":"sum","risk_type_field_enum":"currency","risk_type_field_description":
"Insurance amount"},{"risk_type_field_name":"completion","risk_type_field_enum":"date",
"risk_type_field_description":"Date of completion"}]}
```

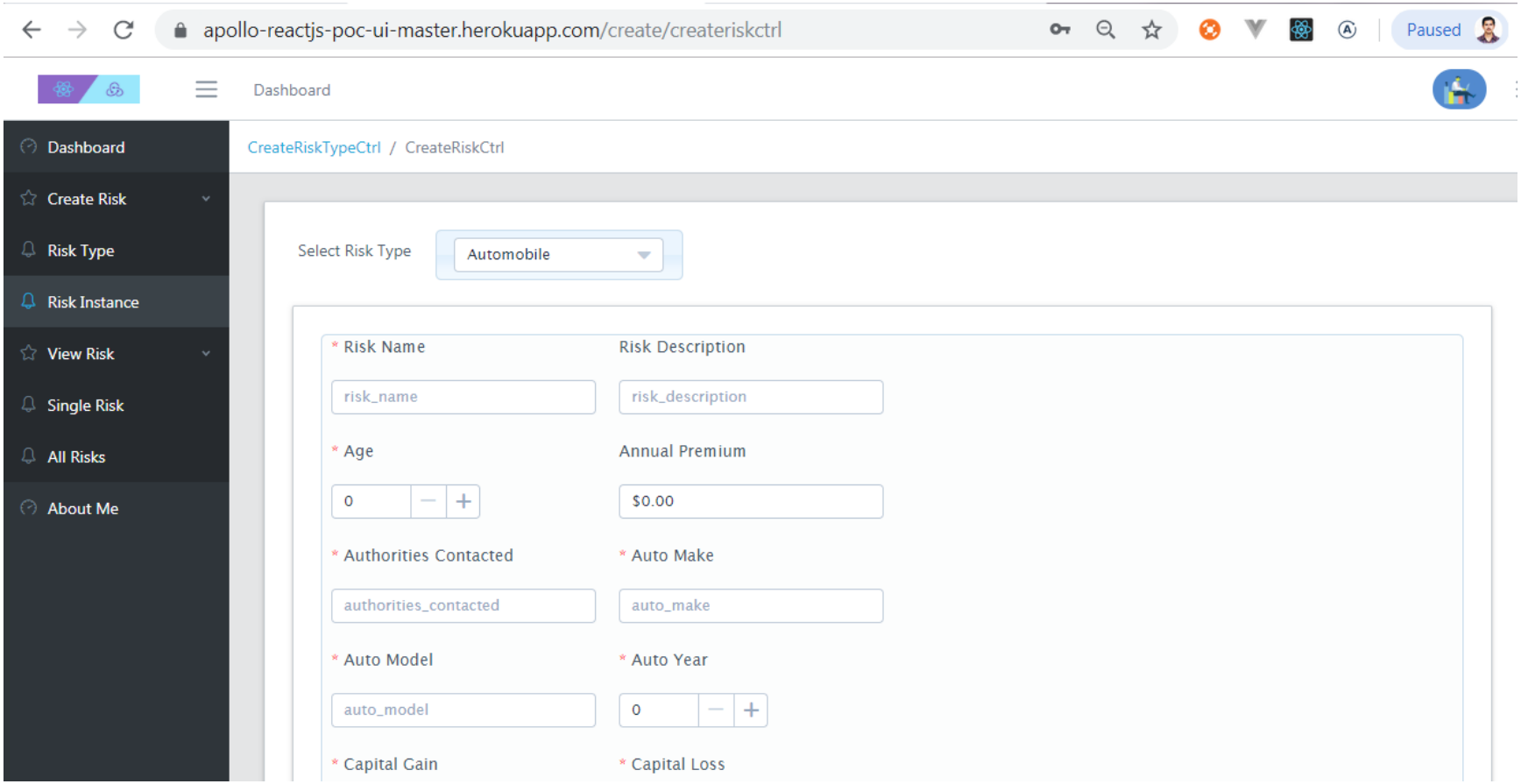
Create RiskType shows Success



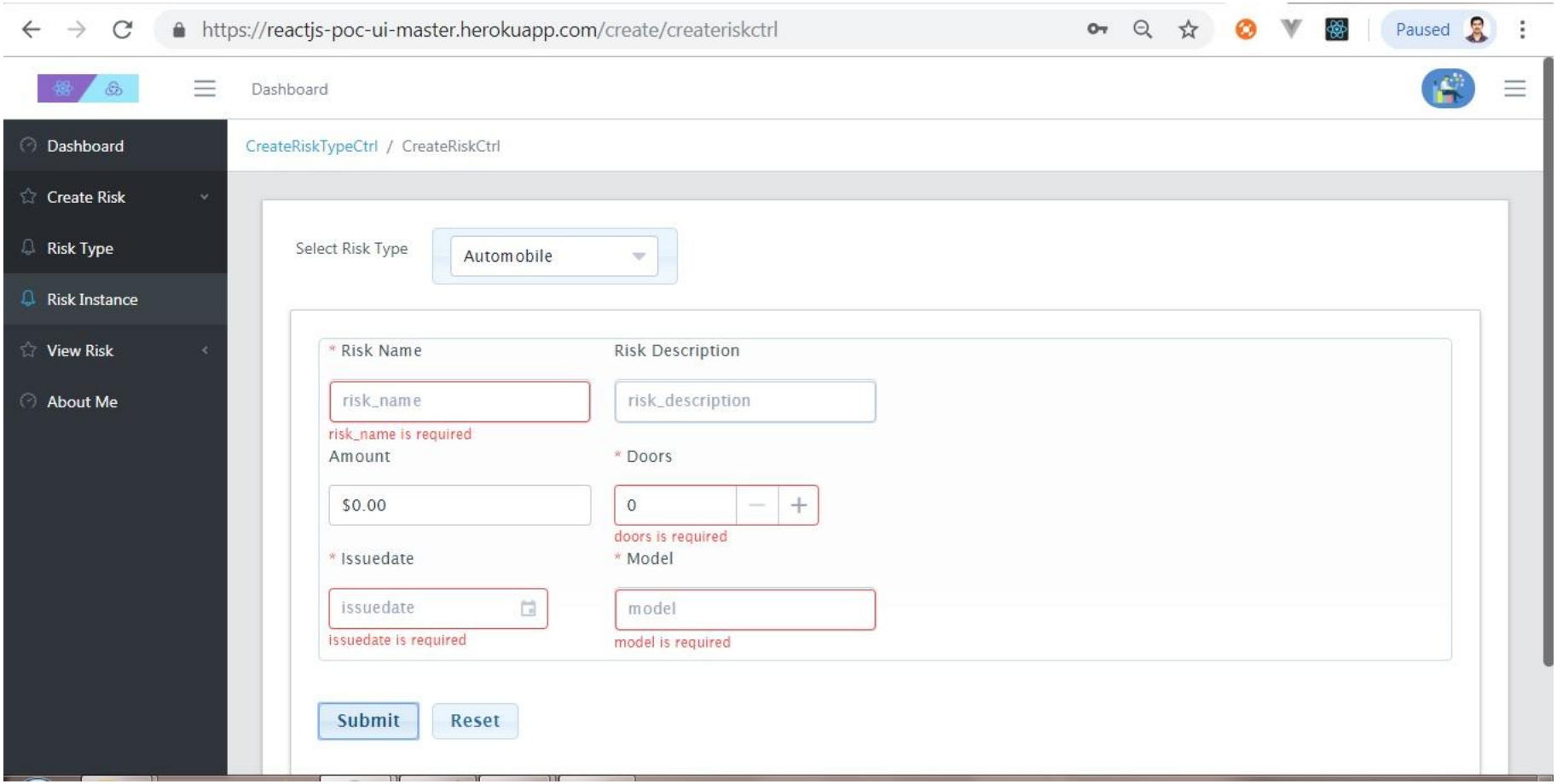
Create Risk Instance screen.

- User can create Risk Instance based on Risk types.
- Select appropriate Risk Type from dropdown box. Screen will fetch Risk type details stored in backend. Dynamically populate appropriate controls based on Entity attributes.
- Fill up all dynamically populated controls that corresponds each Risk Instance field and submit form.
- Only Admin user can define Risk Types.

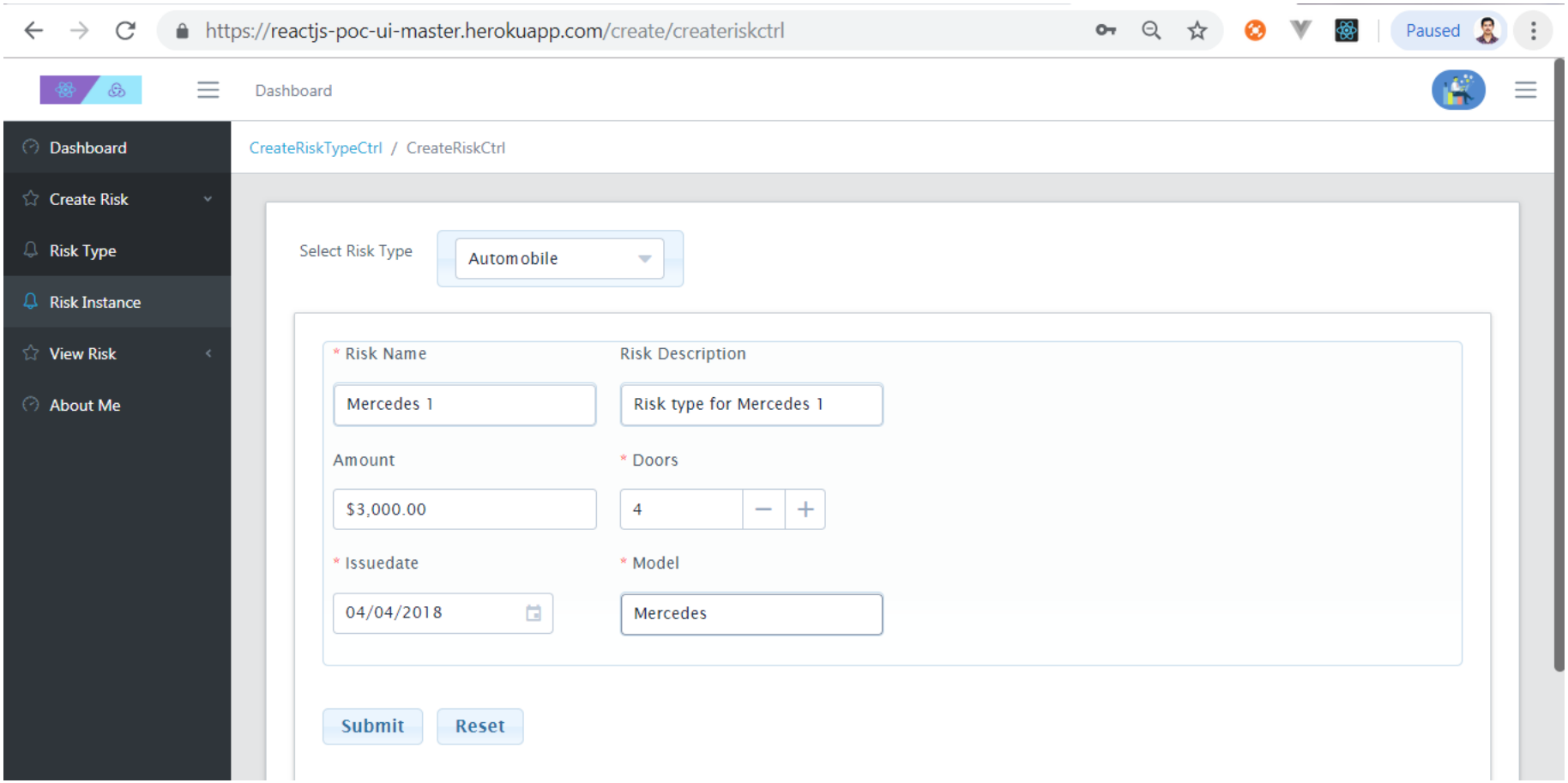
Screen shows dynamically populated controls. Ready to accept input values.



Screen showing field validation errors.



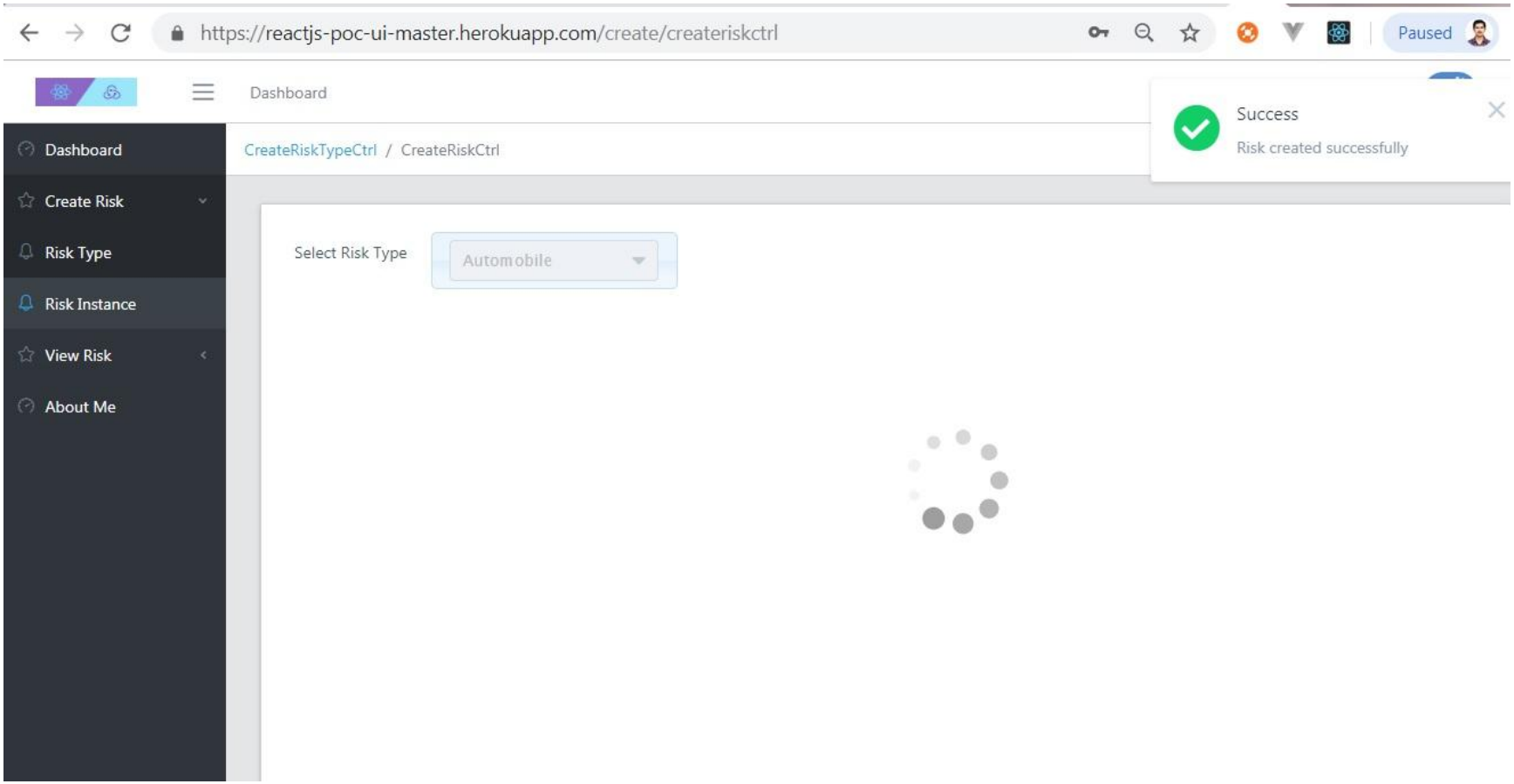
Submit when validation succeed



Posting Risk Instance data to server


```
{"risktype":1,"risk_name":"Mercedes 1","risk_description":"Insurance for Mercedes","risk_riskfields":[{
"risktypefield":4,"risk_field_value":"07/14/2016"},{"risktypefield":3,"risk_field_value":30000},{"risktypefield":
2,"risk_field_value":4},{"risktypefield":1,"risk_field_value":"Mercedes"}]}
```

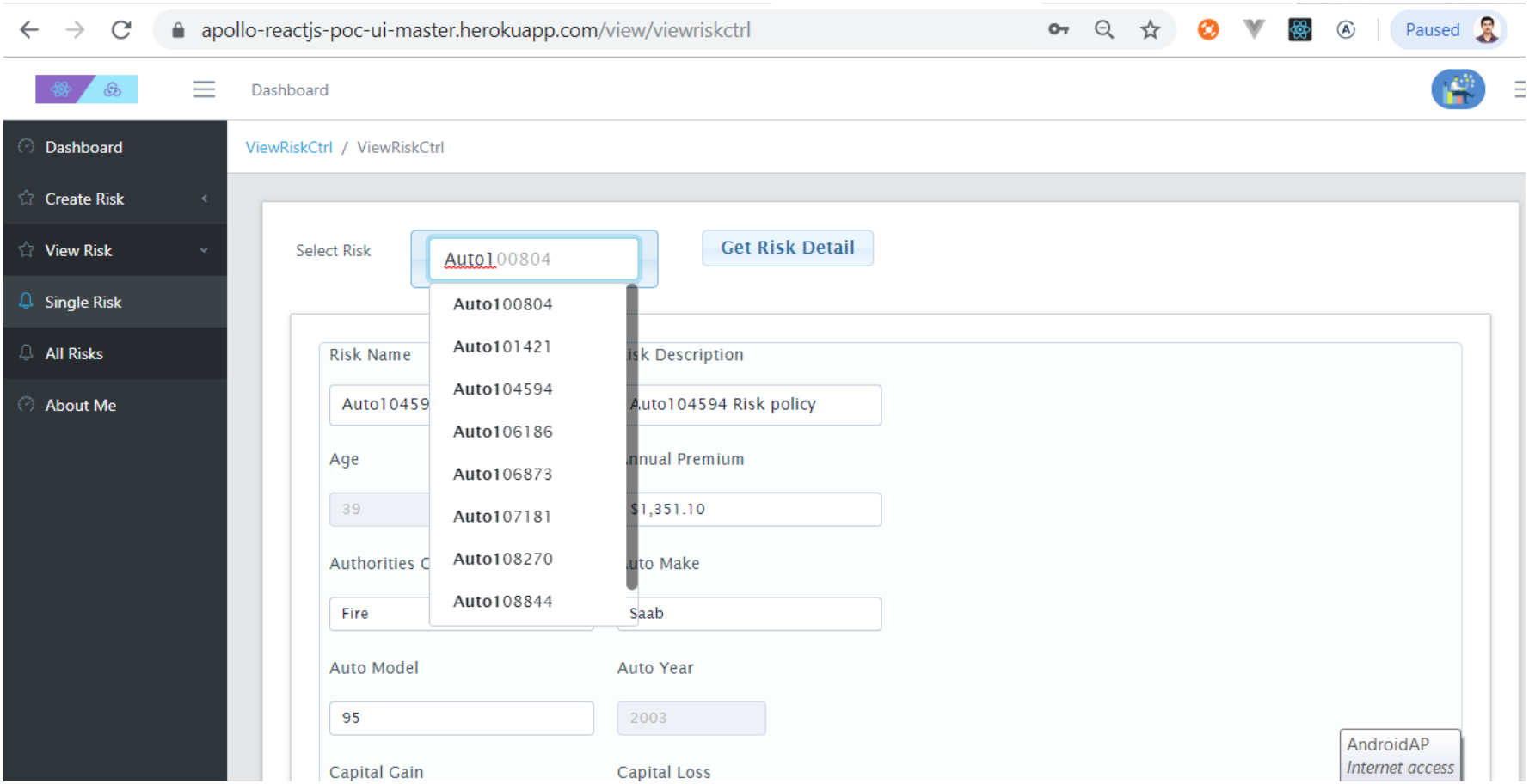
Create Risk Instance showing success



View Single Risk Instance

- User can view single Risk Instance using this form.
- Select appropriate Risk Name from Auto Complete dropdown box.
- Screen will fetch Risk instance details for selected Risk name. Risk details like name and its attributes values are fetched. Screen will dynamically populate appropriate controls based on Entity attributes type also available in response.
- Each dynamic control’s value is set with corresponding entity attribute value.

View Single Risk Screen



View All Risks

- User can view Risk Instances using this form
- Select appropriate Risk Type from dropdown box.
- Screen will show one Page of Risk instances Risk Name and associated instance fields in tabular format for selected Risk Type.
- User can paginate forward and backward through Risk records.

View All Risk Screen

apollo-reactjs-poc-ui-master.herokuapp.com/view/viewriskgrid

Paused

Dashboard

- Dashboard
- Create Risk
- View Risk
- Single Risk
- All Risks
- About Me

ViewRiskCtrl / ViewRiskGrid

Select Risk Type: Automobile

Risk Name	Risk Description	Age	Annual Premium	Authorities Contacted
Auto100804	Auto100804 Risk policy	33	$ 1,459.96	Ambulance
Auto101421	Auto101421 Risk policy	26	$ 1,022.46	Other
Auto104594	Auto104594 Risk policy	39	$ 1,351.10	Fire
Auto106186	Auto106186 Risk policy	25	$ 1,389.86	Other
Auto106873	Auto106873 Risk policy	37	$ 894.40	Police

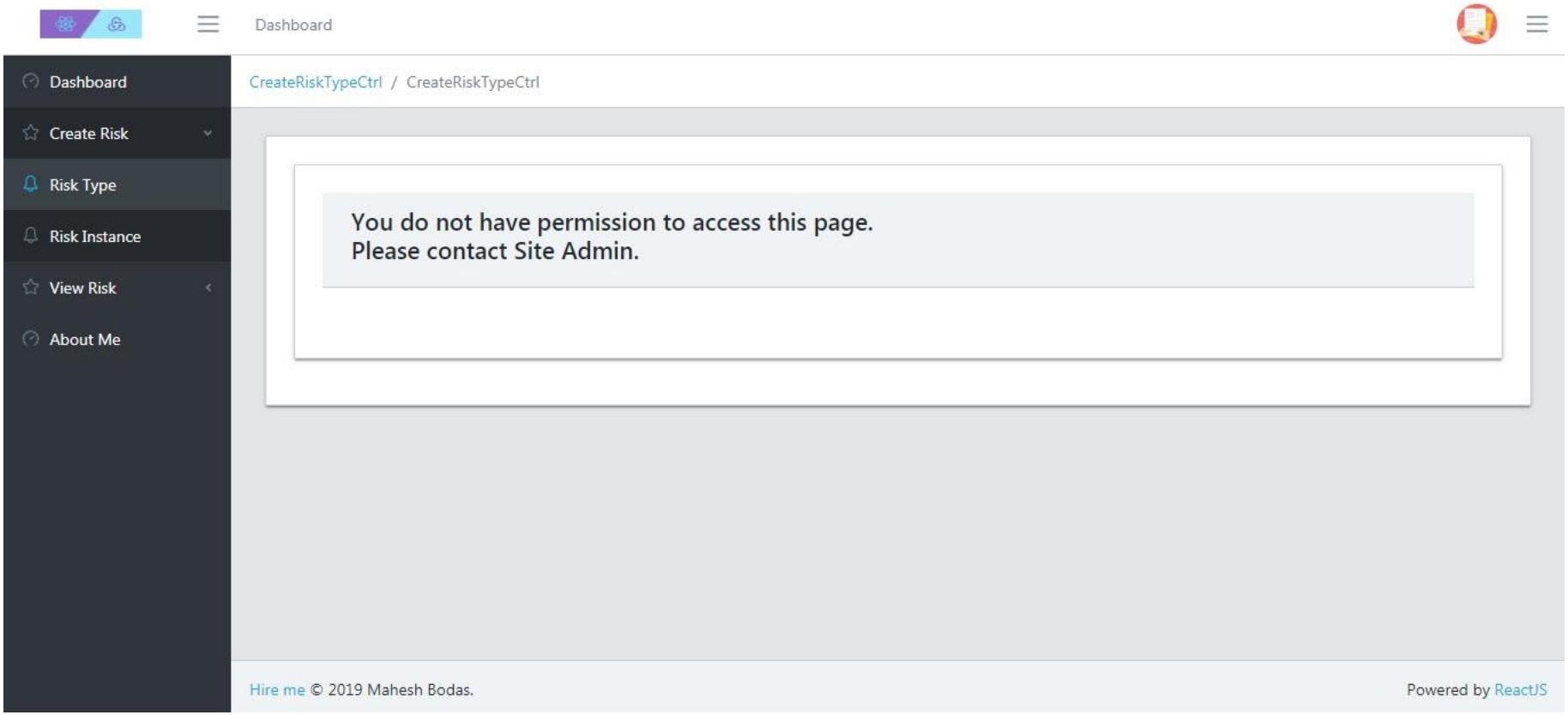
← Previous Page | Next Page →

AndroidAP
Internet access

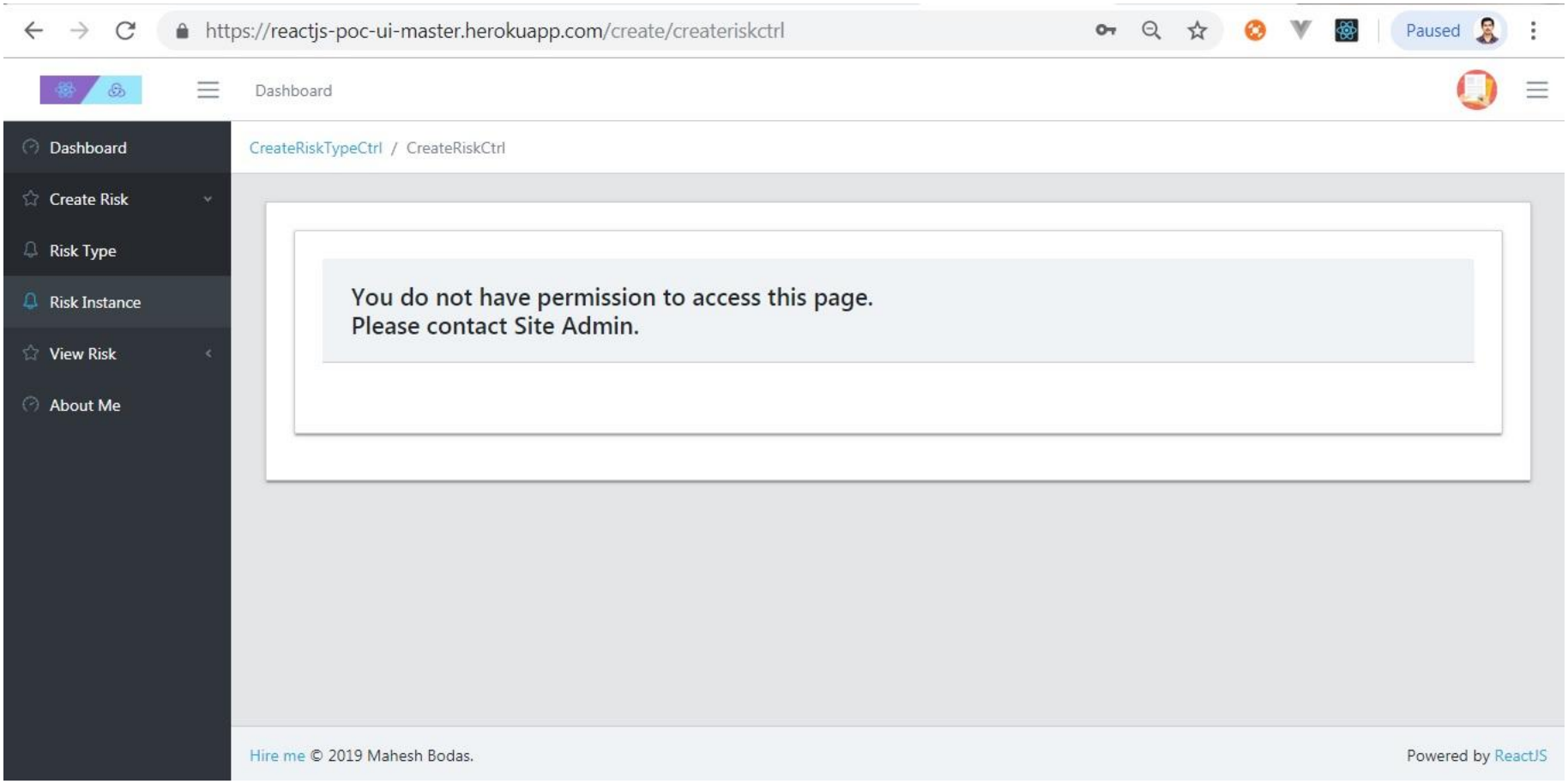
Non Admin users

- All above slides assumed user logged in has admin privilege.
- Non admin user will be restricted from accessing Create Risk Type and Create Risk Instance page.
- They can view Risk Instances added to system.
- User : editor (lower case)
- Password : test#123 (lower case)

No access to Create Risk Type



No access to Create Risk Instance



Unit testing.

- Used Enzyme, Jest for unit testing React components.
- Used Redux Mock Store for Unit testing of, Redux actions, Redux reducers.

Known issues.

- Since Django Rest API and UI application are installed using free account on Heroku platform.
- In case if Django Rest API is not invoked in last one hour or more Dyno associated with it goes to sleep.
- As a result it may be possible that when UI invokes API it might timeout.
- In case if you get any error try refreshing page after 2/3 minutes.

Known issue (continued)